УДК 681.3.06

П.П. Приходько

О ВОЗМОЖНЫХ ОСНОВАНИЯХ НЕМОНОТОННОГО ДЕДУКТИВНОГО СИНТЕЗА ПРОГРАММ

Предлагается подход к практической автоматизации синтеза программ из готовых исполнимых программных составляющих, непосредственно извлекаемых из моделей предметных областей. Этот подход основывается на использовании немонотонных дедуктивных построений. Рассматривается возможность организации действенных баз данных для задач построения конкретных программ.

Введение

Программирование все еще остается довольно кропотливым и трудоемким делом. Программы все еще "пишутся", "кодируются", а не получаются автоматически по "... исходной записи, более близкой к начальной формулировке задачи ... в рамках существенно более широкого языка формулирования задач, не ограничиваемого конкретным классом" [1], как высказывал это А.П. Ершов в 70-х годах прошлого столетия.

Успехи в автоматизации доказательных рассуждений [2] способствовали становлению взглядов о возможности автоматизации дедуктивного синтеза программ по их спецификациям средствами логико-математических исчислений [3, 4]. Установилась точка зрения, в соответствии с которой автоматическое программирование заключается в реализации алгоритма построения (синтеза) требуемой программы из предварительно полученных составляющих, если задача анализа решена положительно, т.е. если такую программу можно построить. Указанные составляющие отвечают условиям задачи, решаемой при выполнении требуемой программы.

Автоматическое программирование предусматривает явное описание в условиях задачи знаний, которые предполагается использовать для построения требуемой программы. В соответствии с этим описанием указываются входные и выходные параметры требуемой программы.

Сложным вопросом в такой организации моделей предметных областей задач автоматического синтеза конкретных программ оказалась эффективность представления объектов, связанных с построением циклических предложений, в частности рекурсивных объектов. В решении этого вопроса получил распространение подход, основывающийся на предположении, что a priori известно о потребности построения соответствующих циклических предложений, т.е. речь идет о собственно программировании циклов [4–6].

Указанную ситуацию можно связать с использованием исчислений, основывающихся на классической логике. Одним из важнейших их свойств является монотонность [7, с. 218]. В силу этого свойства какое-либо утверждение, получаемое в результате логического вывода, не может быть опровергнуто в ходе дальнейших умозаключений. При этом извлечение программы требует хотя бы интенсионального описания всех значений, получаемых в процессе ее выполнения. Предложения программы извлекаются из соответствующих применений правил. Сложности возникают при отождествлении цикличности в последовательностях таких применений.

Этого можно избежать, используя для дедуктивного синтеза программ немонотонное исчисление с эффективной реализацией структурных свойств их предметных областей. Рассмотрим возможный подход к по-

строению такого исчисления, его основания и особенности использования.

1. Основные понятия и определения

Предметные области задач автоматического синтеза конкретных программ имеют глубоко специальный характер. Даже само существование корректно поставленного вопроса о построении программы предполагает, что определены процедурные представления всех исходных отношений, указанных в условиях решаемой задачи. Выполнение этих процедур может понадобиться для получения требуемых значений. Такую предметную область можно эффективно представить вычислительной моделью — совокупностью некоторых исходных функциональных зависимостей, изначально получающих определенную программную реализацию.

Для формализованного представления синтезируемых программ и их предметных областей используем подход, развитый Э.Х. Тыугу [4], А.Я. Диковским и М.И. Кановичем [5, 6] и примененный в [8, 9].

Функции, реализуемые программой, выражают взаимосвязь значений различных объектов, представляемую ее моделью предметной области. Назовем такие объекты программными. Модель предметной области программы заключает в себе всю совокупность знаний об этих объектах. Их значения используются в программе или определяются при ее выполнении.

Программе, описываемой такой моделью, соответствует функциональное отношение, которое связывает значения объектов, представляющих входные и выходные данные. Значения, устанавливаемые для элементов какой-либо совокупности объектов при выполнении программы, можно истолковывать как состояние этой совокупности в указанный момент ее выполнения.

Различные программы могут быть связаны с одной и той же совокупностью объектов, т.е. иметь общую модель предметной области. Такую модель следует рассматривать как совокупность знаний о некотором моделируемом объекте, а получение программы — как установление однозначного соответствия, которое связывает характеристики этого объекта. Его состояние отражается совокупным значением этих характеристик, представляемых программными объектами.

Состояние любого их множества следует рассматривать как часть состояния всех таких объектов, представляющих моделируемый объект. Поэтому модель предметной области можно рассматривать еще и как средство определения состояния моделируемого объекта по значению части его характеристик. В силу наличия взаимосвязей между такими характеристиками всем возможным состояниям такого объекта могут отвечать не все мыслимые значения наборов программных объектов.

Модель предметной области, определяющая требуемую программу, представляет знания, существенные для ее построения, в том числе и знания об алгоритме, реализуемом этой программой. Например, никакая практическая система автоматического синтеза программ "сама" никогда не реализует алгоритм решения большой теоремы Ферма, если для x, y, z и n указать лишь способ их представления в памяти ЭВМ [3].

В [8, 9] составляющие какой-либо предметной области конкретной программы представляются тройкой $(X, \varepsilon, \mathcal{K})$,

где X — множество программных объектов;

$\varepsilon = \{E_x : x \in X\}$ — семейство частично упорядоченных множеств $(E_x, \subseteq_x)_{x \in X}$ этих объектов;

$\mathcal{K}$ — булева алгебра множеств-наборов программных объектов.

Какое-либо состояние набора программных объектов представляет функция вида

$$w : X_w \to \bigcup \varepsilon,$$

где $X_w \in X$ и $w(x) \in E_x$ для $x \in X_W$. Поэтому все такие состояния представляются множеством

$$W \equiv W(X, \varepsilon, \mathcal{K}) = \bigcup \left\{ \left(\prod \varepsilon \right)_C : C \in \mathcal{K} \right\}.$$

Для элементов W можно ввести отношение частичного порядка $\subseteq$. Пусть для $u, v \in W(X, \varepsilon, \mathcal{K})$

$$u \subseteq v \Leftrightarrow \left(X_u \subseteq X_v \,\&\, \forall x \in X_u \;\; u(x) \subseteq_x v(x) \right).$$

Это отношение индуцирует на W операции взятия точных верхней и нижней граней, а также отношения равенства и строгого порядка.

Для множества

$$W_{возм} = W_{возм}\left(X, \varepsilon, \mathcal{K}\right) \subseteq W\left(X, \varepsilon, \mathcal{K}\right) \quad (1)$$

состояний наборов программных объектов, соответствующих возможным состояниям моделируемого объекта, в [8, 9] установлено следующее требование:

$$\forall w \in W_{возм} \forall \mathcal{I} \subseteq \{v : v \in W \,\&\, v \subseteq w\}$$
$$\exists \cup \mathcal{I}, \cap \mathcal{I} \in W_{возм} (\cup \mathcal{I}, \cap \mathcal{I} \in W_{возм} \,\&$$
$$\&\, \cup \mathcal{I}, \cap \mathcal{I} \subseteq w). \quad (2)$$

Пусть функция

$$F : W\left(X, \varepsilon, \mathcal{K}\right) \to W\left(X, \varepsilon, \mathcal{K}\right) \quad (3)$$

каждому состоянию $w \in W$ ставит в однозначное соответствие состояние $Fw \in W$, представляющее все сведения о состоянии моделируемого объекта, получаемые исходя из w на основе знаний о предметной области, заключенных в некоторой ее модели. В [8] показано, что эта функция должна удовлетворять таким условиям:

$$\forall w \in W \; w \subseteq Fw, \quad (4)$$

$$F\left(W_{возм}\right) \subseteq W_{возм} \quad (5)$$

и

$$\forall w \in W_{возм} \quad \forall v \subseteq Fw \quad Fv \subseteq Fw. \quad (6)$$

Там же моделью первого типа предметной области (МПрОI) названо пятерку (X, ε, $\mathcal{K}$; F, $W_{возм}$), где (X, ε, $\mathcal{K}$) — представление составляющих предметной области (ПСПрО); $W_{возм}$ — множество возможных состояний наборов программных объектов, удовлетворяющее условиям (1) и (2), а F — функция вида (3), удовлетворяющая условиям (4) — (6).

Пусть для $A \subseteq X$

$$W(A) \equiv W(X, \varepsilon, \mathcal{K}, A) =$$
$$= \{w : w \in W(X, \varepsilon, \mathcal{K}) \& X_w \subseteq A\},$$
$$W^*(A) \equiv W^*(X, \varepsilon, \mathcal{K}, A) =$$
$$= \{w : w \in W(X, \varepsilon, \mathcal{K}) \& X_w = A\},$$
$$\underline{W}(A) \equiv \underline{W}(X, \varepsilon, \mathcal{K}, A) =$$
$$= \{w : w \in W(X, \varepsilon, \mathcal{K}) \& A \subseteq X_w\},$$
$$W_{возм}(A) \equiv W_{возм}(X, \varepsilon, \mathcal{K}, A) =$$
$$= W(X.\varepsilon, \mathcal{K}, A) \cap W_{возм},$$
$$W^*_{возм}(A) \equiv W^*_{возм}(X, \varepsilon, \mathcal{K}, A) =$$
$$= W^*(X, \varepsilon, \mathcal{K}, A) \cap W_{возм},$$
$$\underline{W}_{возм}(A) \equiv \underline{W}_{возм}(X, \varepsilon, \mathcal{K}, A) =$$
$$= \underline{W}(X, \varepsilon, \mathcal{K}, A) \cap W_{возм}.$$

Процедурные представления отношений, выражающих знания о предметной области, можно соотнести с функциями вида

$$f : W^*\left(A_f\right) \to W\left(B_f\right), \quad (7)$$

где A_f, $B_f \in \mathcal{K}$.

Пусть μ — множество таких функций, представляющих зависимости в некоторой предметной области. В [8] для элементов такого множества установлены следующие требования:

$$\forall f \in \mu \forall w \in \underline{W}_{возм}\left(A_f\right) \exists w \cup f\left(w|_{A_f}\right) \in W_{возм}, \quad (8)$$

$$\forall f, g \in \mu \;\; \forall w \in \underline{W}\left(A_f \cup A_g\right)$$
$$\exists w \cup f\left(w|_{A_f}\right) \cup g\left(w|_{A_g}\right) \in W_{возм}, \quad (9)$$

а также

$$\forall f \in \mu \;\; \forall w', w'' \in W^*_{возм}\left(A_f\right) : w' \subseteq w''$$
$$f\left(w'\right) \subseteq f\left(w''\right). \quad (10)$$

Моделью второго типа предметной области (МПрОII) в [8] названо пятерку (X, ε, $\mathcal{K}$, μ, $W_{возм}$), где (X, ε, $\mathcal{K}$) — ПСПрО, μ — конечное множество функций вида (7), удовлетворяющее условиям (8) – (10), а $W_{возм}$ — множество возможных состояний, удовлетворяющее условиям (1) и (2).

Нетрудно убедиться, что в любой МПрОII условие (9) вытекает из условий (8), (10) и (2): для $w \in \underline{W}_{возм}\left(A_f \cup A_g\right)$ $A_f \subseteq X_w$, поэтому $w \in \underline{W}\left(A_f\right)$ и в силу (8) существует $w \cup f\left(w|_{A_f}\right) \in W_{возм}\left(A_f\right)$, но при этом $A_g \subseteq X_w \subseteq X_w \cup X_{w \cup f\left(w|_{A_f}\right)}$, а $w \cup f\left(w|_{A_f}\right) \in W_{возм}\left(A_g\right)$ и, опять же в силу (8),

$$w \cup f\left(w|_{A_f}\right) \cup g\left(\left(w \cup f\left(w|_{A_f}\right)\right)\Big|_{Ag}\right) \in$$

$$\in W_{возм}, \qquad (11)$$

при этом, поскольку $w|_{A_g} \subseteq \left(w \cup \right.$ $\left. \cup f\left(w|_{A_f}\right)\right)\Big|_{Ag}$, то в силу (10)

$$g\left(w|_{A_g}\right) \subseteq g\left(\left(w \cup f\left(w|_{A_f}\right)\right)\Big|_{A_g}\right)$$

и поскольку

$$w \cup f\left(w|_{A_f}\right) \cup g\left(w|_{A_g}\right) \subseteq w \cup f\left(w|_{A_f}\right) \cup$$

$$\cup g\left(\left(w \cup f\left(w|_{A_f}\right)\right)\Big|_{A_g}\right) \in W_{возм},$$

то существует $w \cup f\left(w|_{A_f}\right) \cup g\left(w|_{A_g}\right) \in W_{возм}$.

В [8] обсуждается эквивалентность МПрОI и МПрОII. Утверждается объективный характер предлагаемых формальных определений. При этом для выражений вида (11) существенно используется монотонность, представляемая условиями (6) и (10). Примененный подход основывается на воззрениях Д. Скотта об эффективной организации данных [10, 11]. Но структурированность данных может отражать структурные свойства не только решаемой задачи, но и методов ее решения.

Функциональность использованных отношений связана с процедурным характером знаний о предметной области, они имеют эквивалентное логическое представление. Конструктивность описания этих знаний предполагает наличие конкретизаций предикатов в таком логическом представлении. Использование в таком представлении сентециальных связок обусловливает структуру булевой алгебры множеств для совокупности наборов программных объектов. Монотонность, выражаемая отношением частичного порядка, установленным для состояний таких наборов, связана с учетом всех вычисляемых значений. В ней проявляется монотонность соответствующего логико-математического исчисления.

Поэтому изложенный подход к организации автоматического синтеза программ не позволяет в полной мере избежать упоминавшихся выше сложностей, связанных с применением логико-математических исчислений, когда необходимо выбирать пути реализации моделируемых свойств в требуемых программах и учитывать особенности такой реализации. Ведь далеко не всегда реальное выполнение программы предполагает сохранение всех промежуточных значений. Более того, именно в переприсваиваниях, в циклических предложениях и вызовах процедур проявляются преимущества автоматической обработки данных.

Указанные особенности иллюстрирует использованный в [9] пример модели предметной области последовательного построения чисел Фибоначчи. Представим такую МПрОII $\left(Y, \tau, \mathcal{V}, \varphi, U\right)$, в которой $Y \overset{df}{=} \{u, v, n.m\}$, $\tau(u) = E_u \overset{df}{=} \mathbb{N}^*$ и $\tau(n) = E_n \overset{df}{=} \mathbb{N}^*$, но $\tau(v) = E_v \overset{df}{=} \mathbb{N}$ и $\tau(m) =$

$= E_m \overset{df}{=} \mathbb{N}$, а $\{m,v\} \in \boldsymbol{\mathcal{Y}}$ и $\{u\} \in \boldsymbol{\mathcal{Y}}$, $\varphi \overset{df}{=} \{f,g\}$, где $A_f \overset{df}{=} \{u\}$, $B_f \overset{df}{=} \{u\}$ и для $w \in \underline{W}_{возм}\left(Y,\tau,\boldsymbol{\mathcal{Y}},\{u\}\right) \subseteq U = W_{возм}\left(Y,\tau,\boldsymbol{\mathcal{Y}}\right)$

$$f\left(w|_{\{u\}}\right)(u) = u_1,u_2,...u_n,(u_{n-1}+u_n),$$

если $w(u) \overset{df}{=} u_1,u_2,...,u_n$, а $A_g \overset{df}{=} \{u,m\}$, $B_g \overset{df}{=} \{v\}$ и для $w \in \underline{W}_{возм}(Y,\tau,\boldsymbol{\mathcal{Y}},\{u,m\})$ $g\left(w|_{\{u,m\}}\right)(n) = u_m$, если $w(u) \overset{df}{=} u_1,u_2,...,u_n$ и $m \le n$, иначе $n \notin X_{g\left(w|_{A_g}\right)}$. При этом для всякого $w \in \underline{W}_{возм}(Y,\tau,\boldsymbol{\mathcal{Y}},\{u\})$ существуют $(w(u))_1 = 1$ и $(w(u))_2 = 1$. В этом примере видно, как строение значений $u \in Y$ представляет рекуррентность соотношений, из которых получаются числа Фибоначчи.

Уточним общую формулировку задач автоматизации программирования с учетом возможных утрат монотонности. Средой поддержки программ (СПП) назовем пятерку (X, ε, $\boldsymbol{\mathcal{X}}$, μ, $W_{возм}$), где (X, ε, $\boldsymbol{\mathcal{X}}$) — ПСПрО, μ — конечное множество функций вида (7) таких, что для каждой функции $f \in \mu$ существуют единственные ее входной и выходной наборы $A_f, B_f \in \boldsymbol{\mathcal{X}}$, удовлетворяющие условиям (7) и (10), причем для B_f установлено его единственное разбиение на наборы $B'_f, B''_f \in \boldsymbol{\mathcal{X}}$, для которых

$$B_f = B'_f \cup B''_f, \tag{12}$$

$$B'_f \cap B''_f = \varnothing \tag{13}$$

и

$$\forall w \in \underline{W}_{возм}(A_f)\ \exists w|_{\overline{B''_f}} \cup f\left(w|_{A_f}\right) \in W_{возм}. \tag{14}$$

Очевидно, МПОП представляет частный случай СПП, когда $B''_f = \varnothing$ для всех f вида (7). Примером определенной выше среды может быть предлагаемая СПП $\left(Y',\tau',\boldsymbol{\mathcal{Y}}',\varphi',U'\right)$, которая, как и рассмотренная выше МПрОП (Y, τ, $\boldsymbol{\mathcal{Y}}$, φ, U), представляет последовательное получение чисел Фибоначчи. Пусть $Y' \overset{df}{=} \{u_1,u_2,u_3,v,m\}$, $\tau'\left(u_j\right) = E_{u_j} \overset{df}{=} \mathbb{N} \cup \{\lambda\}$, $j = \overline{1,3}$ а $\tau'(v) = E_v \overset{df}{=} \mathbb{N}$, $\tau'(m) = E_m \overset{df}{=} \mathbb{N}$, $\varphi' \overset{df}{=} \{f,g,h,r\}$, $A_f \overset{df}{=} \{u_1,u_2\}$, $B_f = B''_f \overset{df}{=} \{u_3\}$ и для $w \in \underline{W}_{возм}\left(Y',\tau',\boldsymbol{\mathcal{Y}}',A_f\right)$ $w \in U' = W_{возм}\left(Y',\tau',\boldsymbol{\mathcal{Y}}'\right)$ $f\left(w|_{A_f}\right)(u_3) = w(u_1) + w(u_2)$, $A_g \overset{df}{=} \{n\}$, $B_g = B''_g \overset{df}{=} \{n\}$ и для $w \in \underline{W}_{возм}\left(Y',\tau',\boldsymbol{\mathcal{Y}}',\{n\}\right)$ $g\left(w|_{\{n\}}\right) = w(n)+1$, $A_h \overset{df}{=} \{u_3,n,m\}$, $B_h = B''_h \overset{df}{=} \{v\}$ и для $w \in \underline{W}_{возм}\left(Y',\tau',\boldsymbol{\mathcal{Y}}',A_f\right)$ $h\left(w|_{A_h}\right)(v) = w\left(u_3\right)$, если $n = m$, иначе $v \notin X_{h\left(w|_{A_h}\right)}$, а $A_r \overset{df}{=} \{u_2,u_3\}$, $B_r = B''_r \overset{df}{=} \{u_1,u_2,u_3\}$, причем для $w \in \underline{W}_{возм}\left(Y',\tau',\boldsymbol{\mathcal{Y}}',A_f\right)$ $r\left(w|_{A_r}\right)(u_1) = w(u_2)$, $r\left(w|_{A_r}\right)(u_2) = w\left(u_3\right)$ и $u_3 \notin X_{r\left(w|_{Ar}\right)}$.

Примем следующие обозначения. Пусть для любой СПП (X, ε, $\boldsymbol{\mathcal{X}}$, μ, $W_{возм}$), функции $f \in \mu$ и для состояния $w \in W\left(X,\varepsilon,\boldsymbol{\mathcal{X}}\right)$

$$F_f \equiv \begin{cases} w|_{\overline{B''_f}} \cup f\left(w|_{A_f}\right), \text{если } w \in \underline{W}_{возм}(A_f), \\ w \text{ для всех остальных } w, \end{cases}$$

а для множества μ $\mu^* \underset{Df}{=} \bigcup\left\{\mu^n : n \in \mathbb{Z}_+\right\}$, $\mu^+ \underset{Df}{=} \bigcup\left\{\mu^n : n \in \mathbb{N}\right\}$, $\mu^0 \underset{Df}{=} \{\lambda\}$, где λ — отсутствие символа, и для цепочки символов $\boldsymbol{f}$ $\boldsymbol{f} \overset{df}{=} \left(f_i,_{i=\overline{1,j}}\right)$ и $|\boldsymbol{f}| \overset{df}{=} j$, а для натуральных m, n $[\boldsymbol{f}]_m \overset{df}{=} \left(f_i,_{i=\overline{m,j}}\right)$, если $m \le j$, и

$$[\boldsymbol{f}]^n \overset{df}{=} \left(f_i,_{i=\overline{1,n}}\right), \tag{15}$$

если $n \le j$, а $[\boldsymbol{f}]_m^n \overset{df}{=} (f_{i,i=\overline{m,n}})$, если $m \le n \le j$, и

$$(\boldsymbol{f})_i \underset{Df}{=} f_i \qquad (16)$$

для натурального $i \le |\boldsymbol{f}|$, а $r(\boldsymbol{f}) \underset{Df}{=} \{f : f = (\boldsymbol{f})_i \,\&\, i \in \mathbb{N} \,\&\, i \le |\boldsymbol{f}|\}$. Пусть при этом для $w \in W(X, \varepsilon, \mathcal{X})$ $G(\lambda, w) \underset{Df}{=} w$, а для $\boldsymbol{f} \in \mu^*$ $G(\boldsymbol{f}, w) \underset{Df}{=} G([\boldsymbol{f}]_2, F_{f_1} w)$ и $G(\boldsymbol{f}, w) \underset{Df}{=} F_{f_n} G([\boldsymbol{f}]^{n-1}, w)$, если $|\boldsymbol{f}| \le 2$, и $G(\boldsymbol{f}, w) \underset{Df}{=} F_f w$ для $\boldsymbol{f} \in \mu$. Для какой-либо цепочки функциональных символов $\boldsymbol{f} \in \mu^*$

$$\underline{A}_{\boldsymbol{f}} \underset{Df}{=} \bigcup \{A_{f_i} : i \in \mathbb{N} \& i \le |\boldsymbol{f}|\},$$

$$\underline{B}_{\boldsymbol{f}} \underset{Df}{=} \bigcup \{B_{f_i} : i \in \mathbb{N} \& i \le |\boldsymbol{f}|\},$$

$$B''_{\boldsymbol{f}} \underset{Df}{=} \bigcup \{B''_{f_i} : i \in \mathbb{N} \& i \le |\boldsymbol{f}|\},$$

$$B'_{\boldsymbol{f}} \underset{Df}{=} B_{\boldsymbol{f}} \setminus B''_{\boldsymbol{f}},$$

$$A_{\boldsymbol{f}} \underset{Df}{=} \bigcup \left\{ A_{f_i} \setminus B_{[\boldsymbol{f}]^{i-1}} : i \in \mathbb{N} \& i \le |\boldsymbol{f}| \right\},$$

причем $\underline{A}_\lambda = A_\lambda = B_\lambda = \varnothing$.

Какое-либо состояние набора программных объектов отражает некоторую часть данных о состоянии моделируемого объекта. Все данные о состоянии такого объекта, соответствующего СПП (X, ε, $\mathcal{X}$; μ, $W_{возм}$), которые можно получить исходя из определенного состояния набора программных объектов $w \in W(X, \varepsilon, \mathcal{X})$, представим как

$$F_\mu \underset{Df}{=} \bigcup \left\{ G(\boldsymbol{f}, w)\big|_{\overline{B''_{\boldsymbol{f}}}} : \boldsymbol{f} \in \mu^* \right\}. \qquad (17)$$

Нетрудно убедиться в справедливости следующего утверждения [8 с. 45], выражающего связь МПрОII и МПрОI.

Теорема 1. Пусть (X, ε, $\mathcal{X}$; μ, $W_{возм}$) — МПрОII. Тогда (X, ε, $\mathcal{X}$; F_μ, $W_{возм}$) — МПрОI, если $\forall w \in W_{возм}\ \exists \boldsymbol{f} \in \mu^*$ $F_\mu \equiv G(\boldsymbol{f}, w)$.

На основе определения СПП задача синтеза программы формулируется следующим образом: для известной СПП (X, ε, $\mathcal{X}$; μ, $W_{возм}$) и указываемых $A, B \in \mathcal{X}$ реализовать функцию $f : W^*(A) \to W(B)$ такую, что $f(w) \equiv (F_\mu w)\big|_B$ для $w \in W^*_{возм}(A)$. Здесь наборы программных объектов представляют значения входных и выходных параметров, функция f — требуемую программу, а СПП (X, ε, $\mathcal{X}$; μ, $W_{возм}$) — ее спецификацию. Очевидно, решение задачи следует считать существующим, если $F_\mu w \in \underline{W}_{возм}(B)$ для всех $w \in W^*_{возм}(A)$.

Следующее утверждение выражает взаимосвязь МПрОII и СПП, а также пути использования СПП.

Теорема 2. Пусть (X, ε, $\mathcal{X}$; μ, $W_{возм}$) — СПП, $A, B \in \mathcal{X}$ и для любого $w \in W^*_{возм}(A)$ существует $F_\mu w$ вида (17) такое, что $F_\mu w \in \underline{W}_{возм}(B)$ и указанному $w \in W^*_{возм}(A)$ соответствует цепочка $\boldsymbol{f} \in \mu^*$, для которой

$$F_\mu w \equiv G(\boldsymbol{f}, w)\big|_{\overline{B''_{\boldsymbol{f}}}}. \qquad (18)$$

Тогда существует МПОII (X, δ, $\mathcal{X}$; ν, V) такая, что $E_x \subset \delta(x)$, если

$$x \in \bigcup \left\{ B''_{\boldsymbol{f}} : \boldsymbol{f} \in \mu^* \,\&\, F_\mu w \equiv \right.$$

$$\left. \equiv G(\boldsymbol{f}, w)\big|_{\overline{B''_{\boldsymbol{f}}}} \,\&\, w \in W^*{}_{возм}(A) \right\}, \qquad (19)$$

и $E_x = \delta(x)$, если $x \in X$ не удовлетворяет условию (19); при этом $W_{возм}(X, \varepsilon, \mathcal{X}) \subset V$ и $W_{возм}(X, \varepsilon, \mathcal{X}, A) = W_{возм}(X, \delta, \mathcal{X}, A)$, $W_{возм}(X, \varepsilon, \mathcal{X}, B) = W_{возм}(X, \delta, \mathcal{X}, B)$; а каждой $h \in \nu$ соответствует $f \in \mu$

такая, что $A_h = A_f$, $B_h = B_f$ и для $w \in W^*_{возм}(A)$ $h(w) \equiv f(w)$; причем для любого $w \in W^*_{возм}(A)$ всякой цепочке $\boldsymbol{f} \in \mu^*$, удовлетворяющей условию (18), соответствует цепочка $\boldsymbol{h} \in \nu^*$, для которой $F_\nu w = G(\boldsymbol{h}, w) \in \underline{W}_{возм}(X, \delta, \mathcal{K}, B)$ и $(F_\nu w)|_B = (F_\mu w)|_B$.

Доказательство состоит в проверке выполнения условий, представляющих заключение теоремы, для пятерки (X, δ, $\mathcal{K}$, ν, V), где $\delta \overset{df}{=} \{D_x : x \in X\}$ и для любого $x \in X$ $(D_x, \leq_x)$ — частично упорядоченное множество, $D_x = E_x^+$, если x удовлетворяет условию (19), и $D_x = E_x$ во всех остальных случаях, а $\leq_x$ совпадает с $\subseteq_x$, если x не удовлетворяет условию (19), иначе для $\boldsymbol{d}', \boldsymbol{d}'' \in E_x^+$

$$\boldsymbol{d}' \leq_x \boldsymbol{d}'' \Leftrightarrow \left(|\boldsymbol{d}'| \leq |\boldsymbol{d}''| \,\&\, \forall k \leq |\boldsymbol{d}'| \boldsymbol{d}'_k \leq_x \boldsymbol{d}''_k\right);$$

$V \overset{df}{=} \bigcup \{V_n : n \in \mathbb{N}\}$, $V_1 \overset{df}{=} W_{возм}(X, \varepsilon, \mathcal{K})$ и для $n \in \mathbb{N}$ $V_{n+1} \overset{df}{=} \{u : u \subseteq w \,\&\, w : X_w \to (\cup\varepsilon)^{n+1} \cup \cup \{(\cup\varepsilon)^j : j = 1,\ldots,n\} \,\&\, [w]^n \in V_n \,\&\, w_{n+1} = F_f w_n \,\&\, f \in \mu\}$, причем для $w \in W(X, \delta, \mathcal{K})$

$$w_n \overset{df}{=} \Lambda x \in X_w . (w(x))_{\min\{n, |w(x)|\}},$$

$$[w]^n \overset{df}{=} \Lambda x \in X_w . [w(x)]^{\min\{n, |w(x)|\}},$$

где Λ — металямбдаабстракция, а $(\ldots)_j$ и $[\ldots]^j$ определены выражениями (16) и (15); множество ν составляют функции, которые соответствуют элементам цепочек $\boldsymbol{f} \in \mu^*$, удовлетворяющих условию (18) для некоторых $w \in W^*_{возм}(A)$, причем для какой-либо из этих функций $h \in \nu$ и соответствующей ей функции $f \in \mu$ для всех $w \in W(X, \delta, \mathcal{K}, Af)$ и для $x \in B_f$, если x удовлетворяет условию (19), то для $n \in \mathbb{N}$ $(h(w)(x))_n = f(w_n)$, иначе $h(w)(x) = \cup\{f(w_n)(x) : n \in \mathbb{N}\}$.

Теорема 2 показывает определенное качественное различие между понятиями МПрОП и СПП. МПрОП более отождествляется с "предметной областью", а СПП — с "проблемной областью". СПП более ориентирована на учет путей решения представляемых задач. В МПрОП знания о предметной области представляются хотя и в процедурной форме, но более общо. Это различие можно рассмотреть, сравнивая предложенные выше МПрОП (Y, τ, $\mathcal{K}$, φ, U) и СПП(Y', τ', $\mathcal{K}'$, φ', U').

2. Планирование действий синтезируемых программ

Еще во времена становления дедуктивного подхода к автоматизации программирования использование алгоритмических языков высокого уровня дало "естественное" толкование организации автоматического синтеза программ [12—13]. Его можно представить следующими положениями.

Положение 1. Исполнимые программные предложения размещаются в тексте программы в очередности, определяющейся наличием требуемых значений их входных параметров.

Положение 2. Текст программы заканчивается, когда получены значения ее искомых выходных параметров.

Положение 3. Исполнимые программные предложения не касаются программы, если они не являются звеньями цепочки исполнимых программных предложений, в которой входные параметры каждого такого звена являются входными параметрами программы или выходными параметрами предыдущих звеньев либо предыдущего звена, а выходные параметры конечного звена содержат хотя бы один выходной параметр программы.

Выполнение любой программы, поддерживаемой в какой-либо СПП (X, ε, $\mathcal{K}$, μ, $W_{возм}$), представляется цепочкой функциональных символов-элементов

μ, соответствующих действиям программы в их очередности. Рассмотрим свойства таких цепочек и условия их рациональной организации.

Пусть для СПП $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$, каких-либо $\boldsymbol{f}, \boldsymbol{g} \in \mu^*$ и $n \in \mathbb{N}$ при $n \le |\boldsymbol{f}|$

$$\boldsymbol{g} \underline{[}_n \boldsymbol{f} \Leftrightarrow \Big(\boldsymbol{g} = \big(f_{m_k}, {}_{k=\overline{1,L}}\big) \& \, m_1 = n \, \& \, L \le |\boldsymbol{f}| - n + 1 \, \& \, \forall k \in \mathbb{N} : 1 < k \le L \big(m_{k-1} < m_k \, \& \, \& \Big(A_{g_k} \cap B_{g_{k-1}}\Big) \setminus B''_{[\boldsymbol{f}]_{m_{k-1}+1}^{m_k - 1}} \ne \varnothing \Big) \& \, \& \, B_{g_L} \setminus B''_{[\boldsymbol{f}]_{m_L+1}} \ne \varnothing \Big).$$

Следующее утверждение представляет условие рациональной организации действий, выражаемых цепочками функциональных символов СПП.

Теорема 3. Пусть $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$ — СПП, а $\boldsymbol{f} \in \mu^*$. Если для натурального $n \le |\boldsymbol{f}|$ не существует цепочки $\boldsymbol{g} \in \mu^*$ такой, что $\boldsymbol{g} \underline{[}_n \boldsymbol{f}$, то

$$G(\boldsymbol{f}, w) \equiv G\Big([\boldsymbol{f}]^{n-1} [\boldsymbol{f}]_{n+1}, w\Big)$$

для любого $w \in W(X, \varepsilon, \mathcal{X})$.

Нетрудно убедиться в справедливости следующего вспомогательного утверждения, которое требуется для доказательства этой теоремы.

Лемма 1. Пусть $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$ — СПП, а $\boldsymbol{f} \in \mu^*$ и $B \in \mathcal{X}$. Если существует натуральное m такое, что $m < |\boldsymbol{f}|$ и $B \subseteq \cup \Big\{ B''_{f_j} : j \in \mathbb{N} \, \& \, j \le m \Big\}$, причем $\forall j \in \mathbb{N} : j \le m \; A_{f_j} \cap B = \varnothing$, то $G(\boldsymbol{f}, w') \equiv \equiv G(\boldsymbol{f}, w'')$ для каких-либо $w', w'' \in \in \underline{W}_{возм}(A_{\boldsymbol{f}})$ таких, что $w'|_{\overline{B}} \equiv w''|_{\overline{B}}$ и $w'|_B \ne w''|_B$.

Доказательство теоремы 3. Переформулируем эту теорему следующим образом. Докажем, что если для некоторого $w \in W(X, \varepsilon, \mathcal{X})$

$$G(\boldsymbol{f}, w) \ne G\Big([\boldsymbol{f}]^{n-1} [\boldsymbol{f}]_{n+1}, w\Big), \qquad (20)$$

то существует цепочка $\boldsymbol{g} \in \mu^*$, для которой $\boldsymbol{g} \underline{[}_n \boldsymbol{f}$. Из (20) следует, что

$$G\Big([\boldsymbol{f}]^m, w\Big) \ne G\Big([\boldsymbol{f}]^{n-1}, w\Big) \qquad (21)$$

и

$$G\Big([\boldsymbol{f}]^k, w\Big) \ne G\Big([\boldsymbol{f}]^{n-1} [\boldsymbol{f}]_{n+1}^k, w\Big) \qquad (22)$$

для всякого натурального k при $n < k$. Но тогда лемма 1 в соответствующей формулировке применима к (21) и (22) и удовлетворяет условиям существования элементов требуемой цепочки.

Теорема доказана.

Рассмотрим несколько утверждений для рациональной организации действий при решении определенных задач.

Пусть для СПП $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$

$$\mathcal{S} \underset{Df}{\equiv} \Big\{ \boldsymbol{f} : \boldsymbol{f} \in \mu^* \, \& \, \forall n \in \mathbb{N} : n < |\boldsymbol{f}| \exists \boldsymbol{g} \in \mu^* \, \boldsymbol{g} \underline{[}_n \boldsymbol{f} \Big\},$$

а для каких-либо $A, B \in \mathcal{X}$ $\mathcal{Z}(A, B) \underset{Df}{=}$

$$\underset{Df}{=} \Big\{ \boldsymbol{f} : \boldsymbol{f} \in \mathcal{S} \, \& \, A_{\boldsymbol{f}} \subseteq A \, \& \, B \subseteq B'_{\boldsymbol{f}} \, \& \, \forall n \in \mathbb{N} : n \le |\boldsymbol{f}| \exists k \in \mathbb{N} : \Big(k \le |\boldsymbol{f}| \, \& \, A_{f_k} \setminus B''_{[f]^{k-1}} \ne \varnothing \Big) \exists \boldsymbol{g} \in \mu^* : \big(\exists m \in \mathbb{N} : f_n = g_m \big) \big(\boldsymbol{g} \underline{[}_k \boldsymbol{f} \, \& \, B'_{\boldsymbol{g}} \cap B \ne \varnothing \big) \Big\}.$$

Следующее утверждение представляет условия, которым удовлетворяет каждый шаг рационально организованных действий.

Лемма 2. Для каких-либо СПП $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$, наборов $A, B \in \mathcal{X}$ и цепочки $\boldsymbol{f} \in \mathcal{Z}(A, B)$

$$[\boldsymbol{f}]_n \in \mathcal{Z}\Big(A \cup B_{[\boldsymbol{f}]^{n-1}}, B \cap B'_{[\boldsymbol{f}]_n} \Big)$$

для любого натурального $n \le |\boldsymbol{f}|$.

Доказательство. Непосредственно из соответствующих определений следует, что $A_{[\boldsymbol{f}]_n} \subseteq A \cup B_{[\boldsymbol{f}]^{n-1}}$ и

$[f]_n \in \mathcal{S}$. Кроме того, очевидно, что $B \cap B'_{[f]_n} \subseteq B'_{[f]_n}$. Остается доказать, что для любого натурального n' при $n \le n' \le |f|$ существует натуральное k такое, что $n \le k \le n'$ и $A_{f_k} \setminus B''_{[f]^{k-1}_{n+1}} \ne \varnothing$, существует цепочка $g \overset{df}{=} \left(f_{q_j}, j = \overline{1,L} \right)$ такая, что для некоторого натурального $m < L$ $q_m = n'$, причем $g \underline{[}_n [f]_n$ и

$$B'_g \cap B \cap B'_{[f]_n} \ne \varnothing. \qquad (23)$$

Из определения $\mathcal{S}$ следует, что существует цепочка $h \overset{df}{=} \left(f_{m_i}, i = \overline{1,t} \right) \in \mu^*$ такая, что $h \underline{[}_n f$, но тогда $k = m_2$, а $g = [h]_2$. Поскольку $f \in \mathcal{Z}(A,B)$, то $B'_h \cap B \ne \varnothing$ и условие (23) удовлетворяется.

Лемма доказана.

Следующее утверждение существенно используется в дальнейшем изложении.

Теорема 4. Пусть $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$ — СПП, для наборов $A, B \in \mathcal{X}$ $\mathcal{Z}(A,B) \ne \varnothing$ и для цепочки $f \in \mu^*$ $f \in \mathcal{Z}(A,B)$. Тогда для какого-либо непустого набора $A' \subseteq A_f$ существует цепочка $g \overset{df}{=} \left(f_{m_i}, i = \overline{1,|g|} \right) \in \mathcal{Z}\left(A_g, B'_g \right)$ такая, что $A' \subseteq A_g \subseteq \underline{A}_f$ и $B'_g \ne \varnothing$, а для какого-либо $w \in \underline{W}^*_{возм}(A)$

$$G(f,w)|_{\overline{B''_g}} \equiv G\left([f]_{m_{|g|}+1}, G(g,u)|_{B'} \cup u|_{\overline{B''_g}} \right)\Big|_{\overline{B''_g}},$$

где $u = G(f', w)$, а $f' \overset{df}{=} \left(f'_j, j = \overline{1, m_{|g|}} \right)$ и для любого натурального $j \le m_{|g|} - 1$ $f'_j = f_j$, если $j \ne m_i$ для всех натуральных $i \le |g|$ и $f'_j = \lambda$, если $j = m_i$ для некоторого натурального $i \le |g|$.

Доказательство. Пусть для какой-либо $h \in \mu^*$ и любого натурального n при $n \le |h|$ $\mathcal{G}(A, A', h) \overset{df}{=} \mathcal{G}\left(A, A', [h]^n \right) \mathcal{G}\left(A \cup \right.$ $\left. \cup B_{[h]_n}, A' \cup B_{\mathcal{G}\left(A, A', [h]^n\right)}, h \right)$ и $\mathcal{G}(A, A', h) \overset{df}{=} h$, если $A_h \subseteq A \cup A'$ и $A_h \cap A' \ne \varnothing$, а если $A_h \setminus (A \cup A') \ne \varnothing$ или $A_h \cap A' = \varnothing$, то $\mathcal{G}(A, A', h) \overset{df}{=} \lambda$ для $h \in \mu$. Требуемую цепочку g представляет выражение $\left[\mathcal{G}(A, A', h) \right]^k$ для натурального $k \le |f|$, если $A' \subseteq A_{\left[\mathcal{G}(A, A', h)\right]^k}$ и $B_{\left[\mathcal{G}(A, A', h)\right]^k} \ne \varnothing$. Такое k существует, поскольку $f \in \mathcal{S}$, а $B'_f \ne \varnothing$, и если бы утверждение теоремы было ложно, то не могло бы существовать и исходной цепочки f — она ведь тоже удовлетворяет требуемым условиям.

Теорема доказана.

Из леммы 2 и теоремы 4 вытекает справедливость следующего утверждения.

Следствие 1. Пусть $(X, \varepsilon, \mathcal{X}, \mu, W_{возм})$ — СПП, для наборов $A, B \in \mathcal{X}$ $\mathcal{Z}(A,B) \ne \varnothing$, а для любого $w \in W^*_{возм}(A)$ существует $f \in \mathcal{Z}(A,B)$, для которой

$$F_\mu w \equiv G(f,w)|_{\overline{B''_f}} \in \underline{W}_{возм}(B). \qquad (24)$$

Тогда для какого-либо возможного состояния $w \in W^*_{возм}(A)$ и соответствующей ему цепочки $f \in \mathcal{Z}(A,B)$, удовлетворяющей условию (24) , для любого непустого набора $A' \subseteq \underline{A}_f$ существует цепочка $g \in \mathcal{Z}(A,B)$ такая, что $A' \subseteq \underline{A}_g \subseteq \underline{A}_f$, $B'_g \ne \varnothing$ и

$$F_\mu w \equiv F_\mu \left(w|_{\overline{B''}} \cup \left(\bigcup \left\{ G(g,u)|_{\overline{B''_g}} : u \in W^*_{возм}(A_g) \right.\right.\right. \&$$

$$\left.\left.\left. \& u \subseteq G([f]^j, w) \& j \in \mathbb{N} \& j \le |f| \right\} \right) \right).$$

Представленное утверждение описывает действия подзадачи, понятие которой в дедуктивном синтезе программ было введено Э.Х. Тыугу [12, 13], развито М.И. Диковским и А.Я. Кановичем [5, 6]. Очевидно, описываемая подзадача является разделяемой, если для указанной цепочки $\boldsymbol{g}$ ее действий $\underline{A}_{\boldsymbol{g}} \subset \underline{A}_{\boldsymbol{f}}$, и независимой, если $\underline{A}_{\boldsymbol{g}} = A'$.

3. Структурные свойства программных составляющих

Представленные выше утверждения основываются на положениях 1—3. Хотя описываемые в них свойства соотносятся с некоторыми необходимыми условиями, примеры, опровергающие их достаточность, в еще большей степени демонстрируют возможности преднамеренного составления неэффективных вычислительных моделей. Перейдем к рассмотрению других структурных свойств исполнимых программных составляющих, позволяющих определить их место в синтезируемых программах.

Следующее утверждение показывает возможность полного использования в синтезируемой программе всех знаний, представляемых в СПП соответствующими функциональными зависимостями.

Теорема 5. Пусть (X, ε, $\mathcal{X}$; μ, $W_{возм}$) — СПП, а для наборов $A, B \in \mathcal{X}$ $\mathfrak{z}(A,B) \neq \varnothing$. Тогда для каких-либо $\boldsymbol{f},\boldsymbol{g} \in \in \mathfrak{z}(A,B)$ существует $\boldsymbol{h} \in \mathfrak{z}(A,B)$, для которой $r(\boldsymbol{h}) = r(\boldsymbol{f}) \cup r(\boldsymbol{g})$.

Доказательство. Сначала рассмотрим условия, которым должна удовлетворять такая цепочка $\boldsymbol{h}$. Из леммы 1 вытекает, что элементы исходных цепочек $\boldsymbol{f}$ и $\boldsymbol{g}$ должны следовать в $\boldsymbol{h}$ в очередности, задаваемой последовательностью множеств $\left(\mathfrak{H}_i, i = 0,1,\ldots\right)$ вида $\mathfrak{H}_i \overset{df}{=} \{(\boldsymbol{n}, j) : (\boldsymbol{n} = 'f' \vee \boldsymbol{n} = 'g') \& A_{\boldsymbol{n}_j} \subseteq A_i \,\&\, \forall i' \in \mathbb{N} : i' < i ((\boldsymbol{n}, j) \notin \mathfrak{H}_{i'} \vee \exists j' \in \mathbb{N} : j' \geq j (A_{\boldsymbol{n}_j} \subseteq A_j \,\&\, A_{\boldsymbol{n}_{j'}} \cap D_i \neq \varnothing \,\&\, A_{\boldsymbol{n}_{j'}} \setminus A_i \neq \varnothing))\}$, где $A_0 = A$, $\mathfrak{H}_0 = \varnothing$, $D_i = A_i \setminus A_{i-1}$, а $A_i = A_{i-1} \cup \left(\cup \left\{ B_{\boldsymbol{n}_j} : (\boldsymbol{n}, j) \in \mathfrak{H}_{i-1} \right\} \right)$. Если в какой-либо момент построения такой цепочки $\boldsymbol{h}$ сначала использовать все элементы $\boldsymbol{f}$, представляемые текущим множеством $\mathfrak{H}_i$, и перейти к таким элементам из $\boldsymbol{g}$, то если для некоторого g_j окажется, что $B_{g_j} \cap A_i \neq \varnothing$, то, в силу теоремы 4, существует цепочка $\boldsymbol{s}$ очередных элементов $\boldsymbol{g}$ такая, что $\boldsymbol{s} \in \mathfrak{z}(A_{\boldsymbol{s}}, B'_{\boldsymbol{s}})$ и $B'_{\boldsymbol{s}} \cap B''_{g_j} = \varnothing$. Поэтому, установив в строящейся цепочке $\boldsymbol{h}$ перед указанным элементом g_j такую цепочку $\boldsymbol{s}$, мы сохраняем на текущем участке $\boldsymbol{h}$ все требуемые свойства. Аналогичные соображения справедливы и когда происходит формирование участка $\boldsymbol{h}$, состоящего из одних элементов $\boldsymbol{f}$.

Теорема доказана.

Пусть для любого СПП (X, ε, $\mathcal{X}$; μ, $W_{возм}$) и наборов $A, B \in \mathcal{X}$

$$\mathfrak{Z}(A,B) \underset{Df}{=} \{\boldsymbol{f} : \boldsymbol{f} \in \mathfrak{z}(A,B) \,\&\, \forall \boldsymbol{g} \in \in \mathfrak{z}(A,B)\, r(\boldsymbol{g}) \subseteq r(\boldsymbol{f})\}.$$

Из теорем 3 и 5 вытекает справедливость следующего утверждения.

Следствие 2. Пусть (X, ε, $\mathcal{X}$; μ, $W_{возм}$) — СПП, а $A, B \in \mathcal{X}$ и для любого $w \in W^*_{возм}(A)$ найдется цепочка $\boldsymbol{f} \in \mu^*$ такая, что существует $F_\mu w \equiv G(\boldsymbol{f}, w)\big|_{\overline{B''_{\boldsymbol{f}}}} \in \in W_{возм}(B)$. Тогда $\mathfrak{Z}(A,B) \neq \varnothing$ и для любого $w \in W^*_{возм}(A)$ существует $\boldsymbol{g} \in \mathfrak{Z}(A,B)$ такая, что $F_\mu w \equiv G(\boldsymbol{g}, w)\big|_{\overline{B''_{\boldsymbol{g}}}} \in \underline{W}_{возм}(B)$.

Пусть для любых СПП (X, ε, $\mathcal{X}$; μ, $W_{возм}$) и наборов $A, B \in \mathcal{X}$

$$\mathfrak{Z}_1(A,B) \underset{Df}{=} \{\boldsymbol{f} : \boldsymbol{f} \in \mu^* \;\&\; \boldsymbol{g} \in \mathfrak{Z}(A,B) \;\&\; r(\boldsymbol{f}) =$$

$$= r(\boldsymbol{g}) \;\&\; |\boldsymbol{f}| = |r(\boldsymbol{f})| \;\&\; \boldsymbol{f} \overset{df}{=} \left(f_j,_{j=\overline{1,|\boldsymbol{g}|}}\right) \;\&$$

$$\&\forall j \in \mathbb{N} : j \le |g| \left(\left(\exists i \in \mathbb{N} : i < j \;\&\; g_i = g_j\right) \Rightarrow\right.$$

$$\left. \Rightarrow f_j = \lambda\right) \;\&\; \left(\left(\forall i \in \mathbb{N} : i < j g_i \ne g_j\right) \Rightarrow f_j = g_j\right)\Big)\Big\}.$$

Из следствия 2 вытекает справедливость следующего утверждения, представляющего необходимое условие положительного решения задачи анализа при дедуктивном синтезе программы.

Следствие 3. Пусть (X, ε, $\mathcal{X}$, μ, $W_{возм}$) — СПП, а $A, B \in \mathcal{X}$ и для любого $w \in W^*_{возм}(A)$ найдется цепочка $\boldsymbol{f} \in \mu^*$ такая, что существует $F_\mu w \equiv G(\boldsymbol{f}, w)\big|_{\overline{B''_{\boldsymbol{f}}}} \in$ $\in \underline{W}_{возм}(B)$. Тогда $\mathfrak{Z}_1(A,B) \ne \varnothing$, причем для любого $\boldsymbol{g} \in \mathfrak{Z}_1(A,B)$ $A_{\boldsymbol{g}} \subseteq A$, $B \subseteq B'_{\boldsymbol{g}}$ и $[\boldsymbol{g}]_n \in \mathfrak{Z}\left(A \cup B_{[\boldsymbol{g}]^{n-1}}, B \cap B'_{[\boldsymbol{g}]_n}\right)$ для любого натурального $n \le |\boldsymbol{g}|$.

4. Исчисление императивных функциональных схем

Логические, а позднее и логико-математические исчисления возникли исторически как средство представления и организации теоретического знания. При попытках автоматизации решения различных задач возникают исчисления "нелогического" типа. Как отмечал С.Ю. Маслов [14, с. 291], моделирование таких исчислений в рамках классической логики может оказаться неадекватным по отношению к решаемым задачам. Также представляется практически нецелесообразным и сведение к логическому исчислению задач автоматизации программирования.

Рассмотрим такое исчисление нелогического типа. Представим формальный язык для описания объектов этого исчисления и его правила, содержащие условия их использования. Объектами этого исчисления являются императивные функциональные схемы. Они представляют императивные программы. Частный случай таких схем — цепочки функциональных символов, представляющие линейные (ациклические) программы. Языком линейных схем является язык кортежей функциональных символов. Обогатим этот язык средствами для представления подзадач и циклов подобно тому, как это выполнено в [8, с. 100] для случая МПрОП. При этом строка функциональных символов становиться одним из нетерминальных символов грамматики.

Пусть 'схема', 'элемент схемы', 'строка', 'функциональный символ', 'цикл', 'подзадача', 'набор', 'абстракция', 'аппликация' — нетерминальные символы этого языка, а $\mu \cup X \cup$ {цикл, для, опр, вып} — множество его терминальных символов, причем (X, ε, $\mathcal{X}$, μ, $W_{возм}$) — СПП. Начальный нетерминальный символ — 'схема'. Синтаксические правила грамматики рассматриваемого формального языка представим в бэкус-науровой форме:

<схема>::=<схема><схема>|<элемент схемы>

<элемент схемы>::=<строка>|<цикл>|<абстракция>|<аппликация>

<строка>::=<строка><функциональный символ><функциональный символ>

<цикл>::=*цикл* <подзадача>

<подзадача>::=*для* <набор>

<абстракция>::=*опр* <подзадача>

<аппликация>::=*вып* <подзадача>

<функциональный символ>::=*f*|*g*|...

<набор>::=*A*|*B*|...|*S*|...

где

$$\mu = \{f, g, \ldots\},$$

$$\mathcal{X} = \{A, B, \ldots, S, \ldots\}.$$

При этом для схем ξ, η и ζ $\underline{A}_\zeta = \underline{A}_\xi \cup \underline{A}_\eta$, $A_\zeta = A_\xi \cup (A_\eta \setminus B_\xi)$, $B_\zeta = B_\xi \cup B_\eta$, $B''_\zeta = B''_\xi \cup B''_\eta$, $B'_\zeta = B_\zeta \setminus B''_\eta$, если $\zeta = \xi\eta$.

Семантика абстракции и аппликации предполагает введение в СПП

подзадачи — программного объекта f с функциональным значением, как это выполнено в [8] для МПрОП. Если $\zeta \equiv \underline{опр}\ \underline{для}\ C$, то $\underline{A}_\zeta = X$, $A_\zeta = \varnothing$, а $B_\zeta = B'_\zeta = \{f(C)\}$. Для $\zeta \equiv \underline{вып}\ \underline{для}\ C$ или $\zeta \equiv \underline{цикл}\ \underline{для}\ C$ наборы A_ζ, $\underline{A}_\zeta$ и B_ζ, B'_ζ, B''_ζ соответствуют наборам A_g, $\underline{A}_g$, B_g, B'_g, B''_g следствия 1.

Семантика рассматриваемого формального языка выражается функцией $H(\zeta, w)$, представляющей значение схемы ζ для исходного возможного состояния *w* подобно тому, как это выполнено в [8, с. 102 – 106].

Очевидно,

$$H(\zeta, w) \equiv G(\zeta, w), \qquad (25)$$

если $\zeta \in \mu^*$;

$$H(\underline{опр}\ \underline{для}\ C, w) \equiv w \cup \Lambda t = f(C).\Lambda\ v \in$$
$$\in W^*_{возм}(C).F_\mu\left(\left(F_\mu w\right)\big|_C \cup v\right); \qquad (26)$$

$$H(\underline{вып}\ \underline{для}\ C, w) \equiv$$
$$\equiv \begin{cases} w \cup (f(C))(w|_C), \text{ если } w \in \underline{W}_{возм}(C \cup \{f(C)\}), \\ w \text{ во всех остальных случаях;} \end{cases} \qquad (27)$$

$$H(\underline{цикл}\ \underline{для}\ C, w) \equiv w \cup (\cup$$
$$\cup \{H(\underline{вып}\ \underline{для}\ C, u) : u \subseteq H(\underline{цикл}\ \underline{для}\ C, w)\}). \qquad (28)$$

Изложенные выше утверждения позволяют представить следующие правила вывода в исчислении императивных функциональных схем:

$$\mathfrak{Z}_1(A, B) \neq \varnothing \,\&\, A_\xi \subseteq A \,\&\, A_\eta \subseteq A \cup B_\xi \,\&$$
$$\&\, \eta \in \mathfrak{Z}(A \cup B_\xi, B \cap B'_\xi) \Rightarrow \xi, \eta \to \xi\eta, \qquad (29)$$

$$\mathfrak{Z}_1(A, B) \neq \varnothing \,\&\, \exists f \in \mathfrak{Z}_1(A, B) \exists m \in \mathbb{N}:$$
$$m \leq |f| A' \subseteq A \cup B_{[f]^{m-1}} \Rightarrow \to \underline{опр}\ \underline{для}\ A', \qquad (30)$$

$$\underline{A}_\xi \cap B_\xi \neq \varnothing \Rightarrow \underline{опр}\ \underline{для}\ A_\xi \cap B_\xi \to$$
$$\to \underline{цикл}\ \underline{для}\ A_\xi \cap B_\xi, \qquad (31)$$

$$A' \subseteq A_\xi \cup B_\xi \Rightarrow \xi\ \underline{опр}\ \underline{для}\ A' \to$$
$$\to \xi\ \underline{вып}\ \underline{для}\ A', \qquad (32)$$

где A, A', B — наборы в СПП (X, ε, $\mathcal{K}$; μ, $W_{возм}$), а ξ, η — императивные функциональные схемы, синтаксис которых описан в бэкус-науровой форме, а семантика — в выражениях (25) – (28). Аксиомами представляемого исчисления являются функциональные символы-элементы μ, рассматриваемые как функциональные схемы — одноэлементные кортежи.

Следует отметить, что наличие выделенных определенных условий использования правил (29) – (32) позволяет получить практический метод построения вывода требуемой императивной функциональной схемы, некоторым образом основываясь на воззрениях С.Ю. Маслова о "башне дедуктивных систем" [15].

5. Организация автоматического синтеза программ с переприсваиваниями

Рассмотрим подход к организации автоматического синтеза программ, основывающийся на использовании баз данных для задач построения конкретных программ и на программной реализации построения выводов в представленном исчислении. Проанализируем для этого существо немонотонности в дедуктивных построениях.

Пусть имеющиеся данные о состоянии некоторого моделируемого объекта рассматриваются как состояние среды программы. Это состояние может представлять конъюнкция логических выражений. Если для описания такого объекта использовать определенную СПП (X, ε, $\mathcal{K}$, μ, $W_{возм}$), то указанное состояние среды представляется некоторым $w \in W_{возм}$. Но обращение к какой-либо функциональной зависимости $f \in \mu$ может вызвать изменение состояния среды, причем этому измененному состоя-

нию, представляемому $w|_{\overline{B_f}} \cup f\left(w|_{A_f}\right)$, соответствует конъюнкция логических выражений, получаемая из исходной не только добавлением новых конъюнктов, но и удалением или изменением исходных. Иными словами, происходит немонотонное изменение состояния среды программы, которое с адекватной сложностью невозможно отразить в рамках классической логики.

Состояния объектов среды связываются отношениями, представляемыми в СПП функциональными зависимостями, которые можно истолковывать как правила изменения состояния среды. Таким образом, СПП представляет собой семантическую сеть (рис. 1). Осуществив редукцию [15]

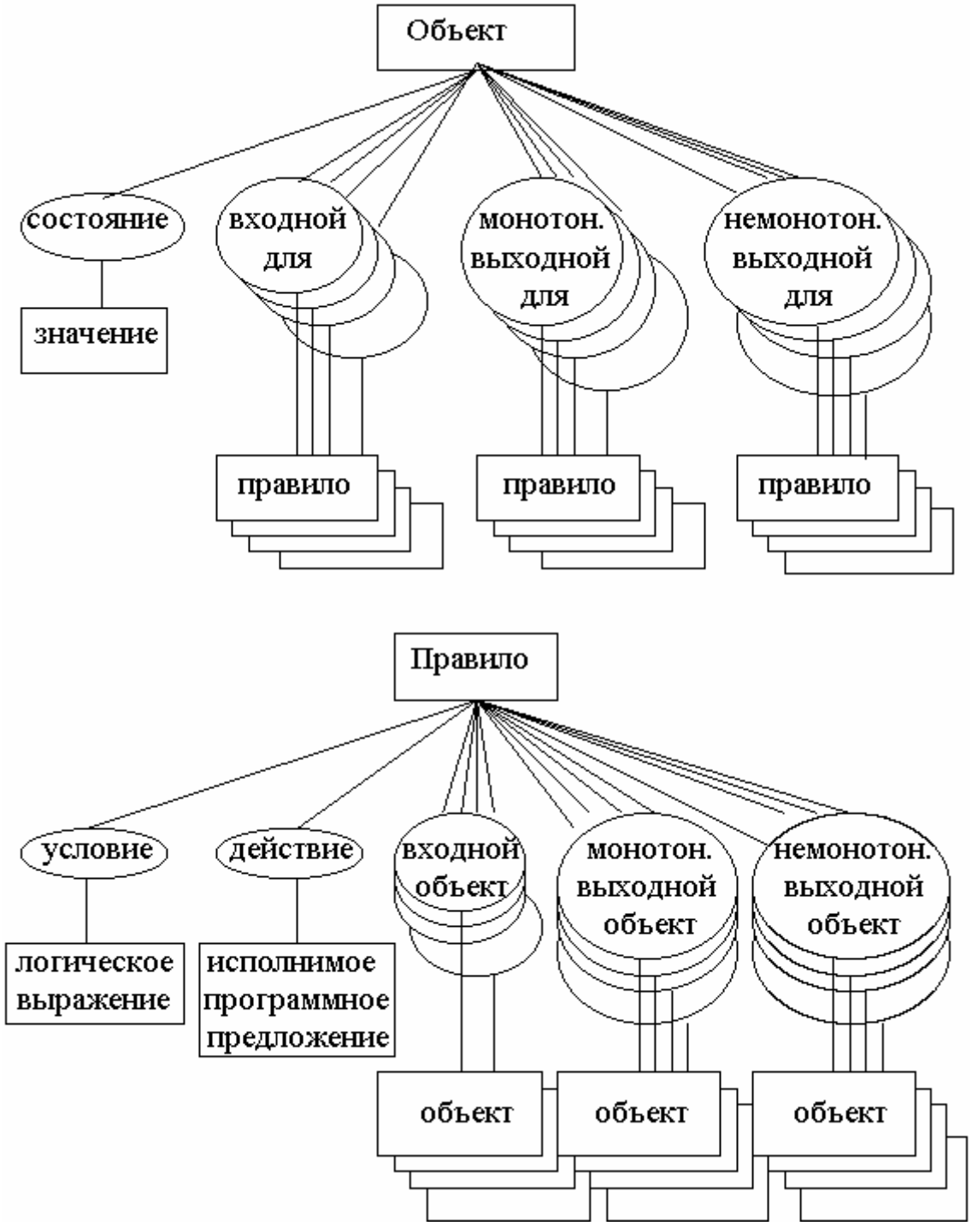

Рис. 1

описанного выше исчисления императивных функциональных схем, можно получить однопосылочное исчисление, объектами которого являются описания состояния среды построения программы. Каждое такое описание отличается от семантической сети СПП лишь тем, что к концептам этой сети типа 'объект' и 'правило' добавляются еще атрибуты, отражающие текущее представление о месте отображаемого объекта, или правила в строящейся программе (рис. 2). Иначе говоря, добавляется еще один параметр со сложным структурированным значением, представляющим релевантность таких концептов требуемой программе.

Данные, во взаимосвязи их получения и использования, при выполнении программы возникают волнами. Начальную волну образуют входные данные программы, а конечная волна должна включать выходные. Но и действия по выполнению команд тоже происходят волнами. Начальную такую волну образуют действия над входными данными. Конечную — действия, завершающие получение выходных данных. Можно говорить о соответствующих волнах объектов и правил среды построения программы. Такие волны можно строить как от начального состояния этой среды (входные волны), так и в обратном направлении — от ее желательного конечного состояния (выходные волны). Критерием релевантности концепта (объекта или правила) программе является его принадлежность к пересечению каких-либо ее входных и выходных волн. Отметим, что приведенные выше рассуждения непосредственно вытекают из следствия 3.

Итак, значение релевантности каждого концепта ‘объект‘ или ‘правило‘ среды построения программы в свою очередь имеет атрибуты ‘номер входной волны‘ и ‘номер выходной волны‘ с целыми неотрицательными необязательными значениями (рис. 3).

В случае ациклической программы или монотонной логической системы синтез программы сводится к построению указанных пересечений

Рис. 2

Рис. 3

волн. Однако как раз состояния построения программы, связанные с переприсваиваниями, с утратой монотонности, позволяют синтезировать составляющие, в которых проявляется ее программистская сущность и достоинства — циклические предложения, проверки, локальные процедуры.

Из следствия 1 вытекает, что всякое немонотонное изменение состояния среды программы для объектов, состояние которых уже имело определенное значение, является циклическим. Участок программы, составляющий тело цикла, начинает образовываться от объектов, состояние которых претерпело такое изменение, до объектов, для которых оговорены только монотонные изменения состояния во всех правилах. Такое образование участка происходит в пределах общих волн объектов и правил (пересечений их входных и выходных волн).

Так образуется тело цикла-рекурсии, если упомянутое немонотонное изменение состояния объектов вызывается выполнением правила, принадлежащего этому же участку. Ес-

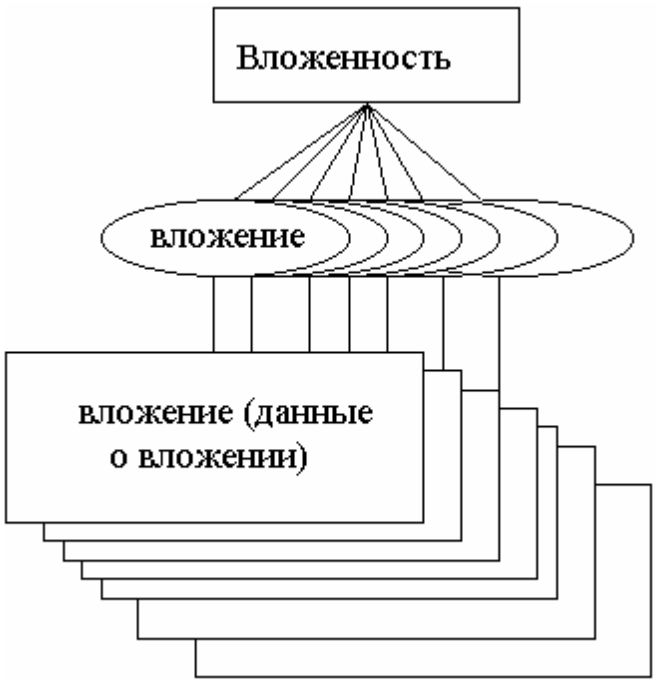

Рис. 4

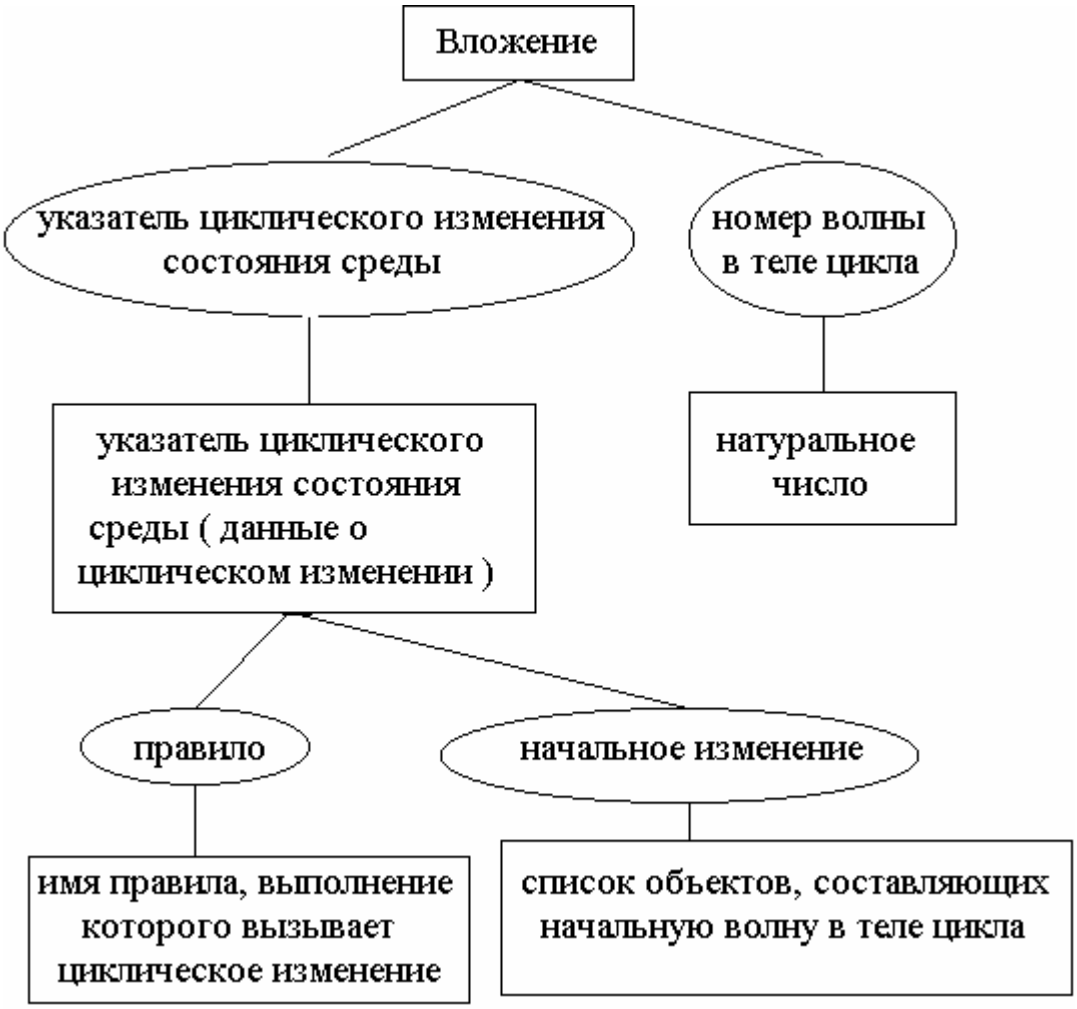

Рис. 5

ли же немонотонное изменение состояния вызвано существованием нескольких независимых правил, обладающих общим выходным объектом (или объектами), для которого (или для которых) возможны немонотонные изменения состояния, то имеет место итерация.

Следует учесть возможность существования монотонных циклических изменений, рассмотренную в [14]. Монотонное изменение состояния среды программы для объектов, состояние которых уже имело определенное значение, является циклическим, если оно вызвано применением правила, состояние входных объектов (объекта) которого определяется состоянием вышеуказанных объектов. Иными словами, очередность, определяемая для этого правила номерами входной и выходной волн, предполагает по крайней мере для одного его выходного объекта номер входной волны не более номера входной волны самого правила.

Итак, параметр ‘релевантность‘ должен иметь еще один атрибут — ‘вложенность‘ (рис. 3). Значение этого атрибута представляет отношение соответствующего концепта среды построения программы к ее циклическим предложениям (рис. 4, 5).

Из вышеизложенного следует, что рассматриваемое исчисление состояний среды построения программ содержит правила вывода для определения очередности в тексте программы, циклических изменений, очередности в пределах тел циклов.

Процесс построения вывода конечного состояния среды построения требуемой программы $\mathbf{\Pi}_k$ из ее начального состояния $\mathbf{\Pi}_0$ в этом исчислении имеет вид

$$\Pi_0 \rightarrow \ldots \quad \ldots \quad \ldots \rightarrow \Pi_i \rightarrow \ldots \quad \ldots \quad \ldots \rightarrow \Pi_j \rightarrow \ldots \quad \ldots \quad \ldots \rightarrow \Pi_k$$

определение принадлежности и очередности расположения в тексте программы | определение циклических изменений | определение принадлежности и очередности расположения в циклах

Начальное состояние Π_0 представляет условия решаемой задачи, определенные для модели ее предметной области. Требуемая программа извлекается из конечного состояния Π_k среды ее построения.

Таким образом, при автоматическом синтезе конкретной программы с переприсваиваниями по условиям поставленной задачи строится описание среды в форме семантической сети, представленной на рис. 1. Среда построения программы получается из этой модели предметной области введением параметра ‘релевантность‘ всем определенным в этой модели объектам и правилам. В начальном состоянии определены только начальная (первая) входная и начальная (первая) выходная волны объектов. Начальную входную волну образуют объекты, соответствующие входным параметрам требуемой программы, а начальную выходную — объекты, соответствующие ее выходным параметрам. Поэтому в начальном состоянии среды построения программы в параметре ‘релевантность‘ определены только значения номера входной волны (равные единице) для объектов, соответствующих входным параметрам программы, и значения номера выходной волны (тоже равные единице) для объектов, соответствующих выходным параметрам программы.

При выполнении правила определения очередности в тексте программы получают значения номера входных и выходных волн для объектов и правил изменения состояния среды программы. Тем самым устанавливается принадлежность соответствующих исполнимых программных предложений требуемой программе и порядок их следования в тексте. Одновременно решается задача анализа: если конечная входная волна не содержит объектов, соответствующих выходным параметрам требуемой программы, или конечная выходная волна не пересекается с объектами, соответствующими входным параметрам требуемой программы, то не существует решения задачи, которое должно получаться при выполнении требуемой программы, и такая программа не может быть построена для представленной модели предметной области. Если задача анализа решается положительно, то текст требуемой программы будет включать исполнимые программные предложения, соответствующие правилам изменения состояния среды, для которых получены одновременно номера входной и выходной волн в очередности, определяемой порядком увеличения номера входной волны.

При выполнении правила определения циклических изменений для вывода в исчислении состояний СПП происходит построение списка указателей циклических предложений, принадлежащих требуемой программе. Для всех правил изменения состояния среды программы осуществляется проверка условий, связанных с определением атрибута ‘вложение‘. Формируются наборы соответствующих циклически изменяемых объектов, принадлежащих одной входной волне.

При выполнении правила определения очередности в пределах тел циклов для вывода в рассматриваемом исчислении проверяется очередность следования объектов и правил от таких наборов до объектов, состояние которых изменяется монотонно. Такие правила представляют исполнимые программные предложения, располагающиеся в соответствующих телах циклов в определяемой таким образом очередности. Вложений может быть несколько, поскольку циклы могут вкладываться.

Программа извлекается из конечного состояния среды ее построе-

ния в соответствии с очередностью следования исполнимых программных предложений и их принадлежностью телам циклов. В указанной очередности предварительно отрабатываются релевантные объекты — им в программе соответствуют описания (начальная входная волна объектов) и управляющие предложения (начальной волне объектов каждого тела цикла отвечает предложение начала цикла, конечной — конца).

Выводы

Представляется обоснованной возможность автоматического синтеза каких-либо практически исполнимых программ, т.е. программ с переприсваиваниями. Такой синтез можно рассматривать как поддерживаемую специальными инструментальными средствами технологию программирования в существующих средах на существующих языках.

За рамками настоящей статьи осталось несколько важных и интересных вопросов:

не затрагивались общие вопросы синтеза алгоритмических описаний планов решений, рассматриваемые в работах Ю.В. Капитоновой, А.А. Летичевского и др. (см., напр., [16]);

не рассматривалась интересная возможность использования понятия фиксированной точки [10, 11];

представляет интерес изучение возможностей использования выразительных средств экспликативного и композиционного программирования, обсуждение семиотических аспектов формирования немонотонных вычислительных моделей (состояние объекта среды программы можно истолковывать как его значение, а его соотношения со смежными в семантической сети СПП правилами — как смысл и в этой связи открываются большие возможности для приложения композиционных и экспликативных воззрений [17, 18]);

представляется эффективным развитие теоретико-модельных подходов к изучению немонотонных дедуктивных построений для автоматического синтеза программ, в частности в связи с вопросами синтеза устойчивых вычислительных алгоритмов. Эти вопросы требуют отдельного дальнейшего обсуждения.

1. *Ершов А.П.* Автоматизация программирования // Математическая энциклопедия / Под ред. И. М. Виноградов (глав. ред.) и др. Т. 1. — М.: Сов. энцикл., 1977. — С. 58 — 59.
2. *Чень Ч., Ли Р.* Математическая логика и машинное доказательство теорем. — М.: Наука, 1983. — 360 с.
3. *Лавров С.С.* Синтез программ // Кибернетика. — 1982. — № 6. — С. 11 — 16.
4. *Тыугу Э.Х.* Концептуальное программирование. — М.: Наука, 1984. — 256 с.
5. *Диковский А.Я., Канович М.И.* Вычислительные модели с разделяемыми подзадачами // Изв. АН СССР. Техн. кибернетика. — 1985. — №5. — С.36 — 59.
6. *Канович М.И.* Логические основы синтеза схем программ для решения вычислительных задач // Там же. — 1988. — №2. — С. 82 — 93.
7. *Логический* подход к искусственному интеллекту: от классической логики к логическому программированию: Пер. с франц. / А. Тейз, П. Грибомон, Ж. Луи и др. — М.: Мир, 1990. — 432 с.
8. *Приходько П.П.* Функциональный подход к концептуальному программированию // Дис. ... канд. физ.-мат. наук. — Киев, 1991. — 177 с.
9. *Приходько П.П.* О возможном подходе к организации синтеза программ для структурированных данных // УСиМ. — 1992. — № 3/4. — С.70 — 74.
10. *Скотт Д.* Набросок математической теории вычислений // Кибернет. сб. Нов. сер. — М.: Мир, 1977. — Вып. 14. — С. 107 — 121.
11. *Скотт Д.* Теория решеток, типы данных и семантика // Введение в языки программирования. Абстракция и типология: Сб. ст. — М.: Мир , 1982. — С.25 — 53.
12. *Тыугу Э.Х.* Решение задач на вычислительных моделях // ЖВМ и МФ. — 1970. — Т. 10, № 3. — С. 716 — 733.
13. *Тыугу Э.Х.* Решатель вычислительных задач // Там же. — 1971. — Т. 11, № 4. — С. 992 — 1004.
14. *Маслов С.Ю., Минц Г.Е.* Теория поиска вывода и обратный метод: Дополнение // Чень Ч., Ли Р. Математическая логика и машинное доказательство теорем. — М.: Наука, 1983. — С. 291 — 326.
15. *Маслов С.Ю.* Теория дедуктивных систем и ее применения. — М.: Радио и связь, 1986. — 136 с.
16. *Об использовании* систем уравнений над структурами данных для спецификации и

синтеза программ / Ю.В. Капитонова, А.А. Летичевский, С.П. Горлач, Г.Н. Горлач // Кибернетика. — 1989. — №1. — С. 19—29.
17. *Редько В.Н.* Композиционная структура программологии // Кибернетика и системный анализ. — 1998. — № 4. — С. 47—60.
18. *Редько В.Н.* Экспликативное программирование: ретроспективы и перспективы // Тр. I Междунар. науч.-практ. конф. по программированию. — Киев, 1998. — С. 3—24.

Получено 18.03.03

Об авторе

Приходько Павел Петрович,
канд. физ.-мат. наук, старший научный сотрудник

Место работы автора:
Институт программных систем НАН Украины,
просп. Академика Глушкова, 40,
Киев-187, 03680, Украина
Тел. (044) 532 4690